

Assignment Presentation

CNAE – Cloud Native Architecture & Engineering



Prof. Stefan Tai
Sebastian Werner, Elias Grünewald, Nicola Leschke, Maria Borges

Information Systems Engineering
TU Berlin

Important Dates

Deadlines:

- Registration: **15/05/2023, 23:59 on Moses (MTS)**

Portfolio examinations will take place on:

- **13 + 20 + 23/06/2023**: presentations
- **03/07-14/07**: Oral consultations
- **25/08/2023**: Hand in reports

Examination in
presence.
Physical presence is
required!



Grading

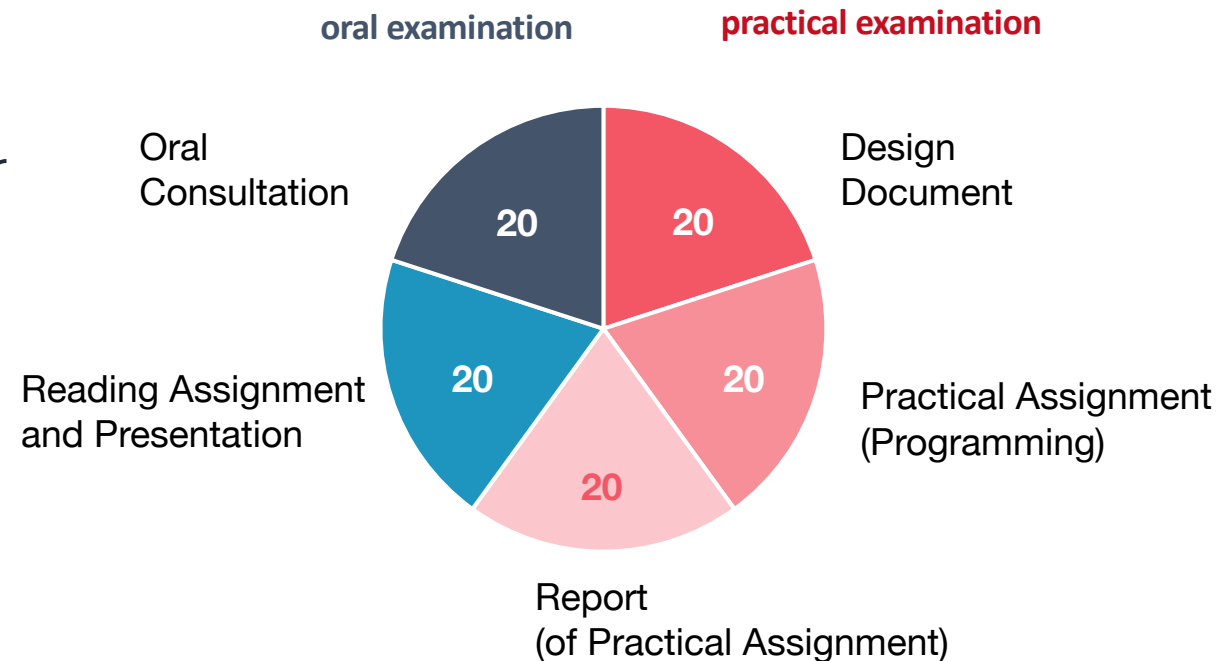
Maximum points: 100 („Portfoliopunkte“)

Points are based on our assessment of your work (and presentations)

Individual grading, also for **group work**
5 members per group

Grading Scale:

1	1,3	1,7	2,0	2,3	2,7	3,0	3,3	3,7	4,0	5,0
95	90	85	80	75	70	65	60	55	50	<50



Assignment Structure

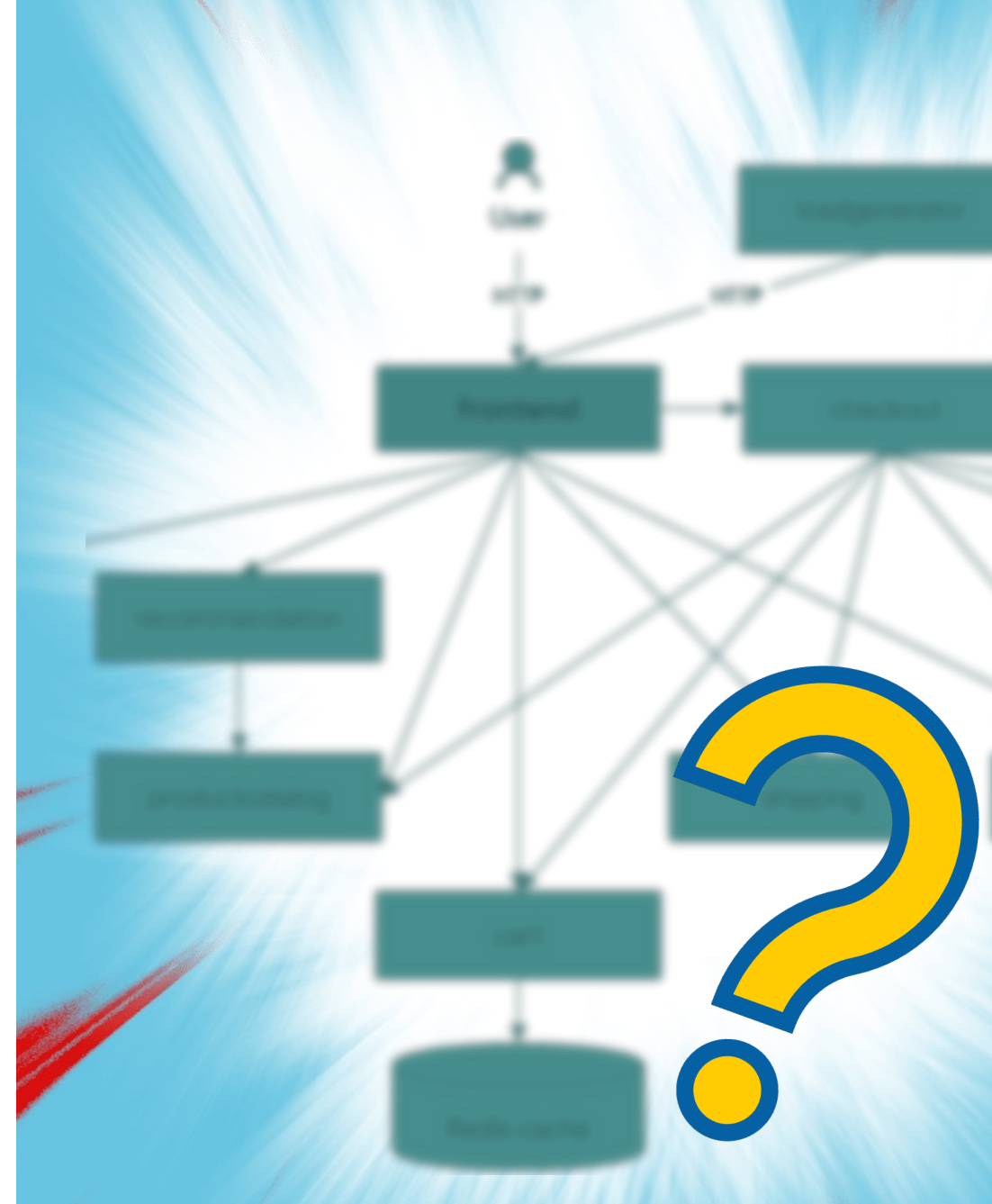
Each group gets one existing “example” architecture and a refactoring goal.

Architecture:

- Always with multiple services
- Consisting of multiple technologies
- May contain old code, old tools or old concepts

Refactoring Goal:

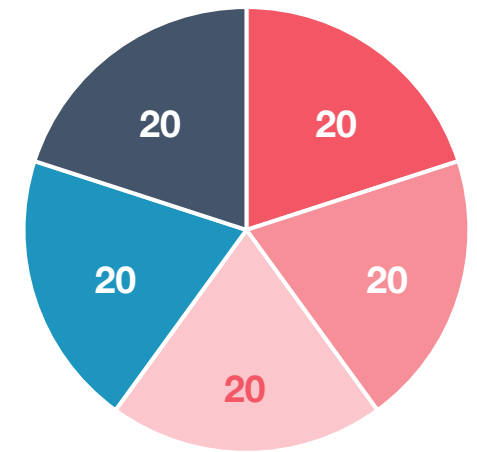
- Improving or controlling a system quality, e.g., robustness



Assignment Elements

There is no clear, correct answer. Instead, you must convince us (lecturers and fellow students) that your plan makes sense.

1. Define the goal and approach [Reading Assignment/Presentation]
2. Write down and document what you are doing and why you are doing it [Design Document]
3. Implement and measure your refactoring approach in realistic environments [Programming]
4. Write it all down [Report]



Reading Assignment and Presentation (1/2)

Content

- A clear understanding of how you define the goal
- A proposal on how you want to address this issue
- An overview of related work as a foundation of your approach

Hints

- We will be strict on time and cut you off after 10 min
- We don't want you to be repetitive. That's why, we will give a brief introduction on each refactoring goal so all groups can start directly with their approaches.

Reading Assignment and Presentation (2/2)

Expected Artifacts:

- 10 min presentation
- 10 min discussion
- Each team must upload the slides as a PDF to ISIS
- **DEADLINE: 12/06/2023, 10 AM**
- We assign the presentation slots afterwards
- Not all of you need to speak, but highlight who did what on the presentation

Design Document (1/4)

Content

- Briefly sketch and motivate the refactoring goal you are about to address with your semester subject
- Describe the particular system and the state of the refactoring goal in the given system architecture
- Identify how you can measure and evaluate this quality goal in the given architecture

Design Document (2/4)

- ...
- Identify and present the state of scientific discussions regarding the refactoring goal / technical principals
- Identify and technically describe available cloud-native technologies that might be used (if any)
- Pay attention to reasonable business and use case assumptions when selecting an approach
- Provide a (sketchy) outlook about what you are going to implement

Design Document (3/4)

- ...
- We want you to explicitly use generative AI-tools to aid you in writing this report. But always show us your own work. Include prompts and were you used AI assistance (e.g., by using a different text color or adding labels).
- However, you are completely responsible for the correctness of the content, so make sure to address any “hallucination” issues (and striking errors) of these tools.

Design Document (4/4)

Expected Artifacts:

- 4-6 Pages incl. references
- Format: IEEE Double Column <https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhnncsfqn>
- 25%-50% generative elements
- Appendix: “creation log” – prompts you used, output you got and where you used the output in the text
- Append a brief description of who contributed what to the document
- **DEADLINE: 12/06/2023, 23:59 PM** upload via ISIS

Implementation

- accessible TU Gitlab repo (`git.tu-berlin.de`)
 - one branch (called **vanilla**) must contain a deployable version of the system before the refactoring
 - **main** branch must contain the refactored version, including necessary documentation
- must be reproducible and transparent:
 - provide **all** build files, helm charts, or IaC/cloud formation files that are necessary
 - provide thorough **deployment instructions in the README**
 - a document describing who did what on the project
- Upload a zipped version of the repo to ISIS
- **DEADLINE: 25/08/2023**

Report (1/2)



- Describe the refactoring steps tried, including the motivation behind them
- Explain the architectural changes in comparison to the original version
- **Evaluate your result in a convincing manner**, showing how you achieved the refactoring goal
- Provide feedback on the approach and general applicability in other systems

Report (2/2)

- 6-8 Pages + References
- Format: IEEE Double Column
- Report can reuse parts of the design document (not more than 2 pages reuse)
- Append a brief description of who contributed what to the document
- **DEADLINE: 25/08/2023, 23:59 PM** PDF upload via ISIS
- Also upload measurement results, evaluation and plotting code, either using GitLab or by providing a **tubCloud** (tubcloud.tu-berlin.de) link, including:
 - raw measurement data
 - printing scripts/notebooks
 - evaluation code, e.g., workload files

Where to run all of this?

All projects are intended to be operated in a cloud environment; thus, the final evaluation should be on a public cloud provider or sufficiently sized cluster.

1. Cloud Providers

1. Limited Credits (50 Euro per student) available on Google Cloud.
Contact werner@tu-berlin.de to get a coupon link.
2. Free tier at Azure 100 Euro per student per year at:
<https://azure.microsoft.com/en-us/free/students/>
3. AWS free tier (needs a credit card)

2. ISE Kubernetes Cluster (Limited resources and timeslots). Contact werner@tu-berlin.de

3. DIY <https://github.com/kubernetes-sigs/kubespray>

Assignment Process

We are now going to present each architecture and goal pair in turn.

To “catch” yourself an assignment you need to ...

1. Find a group of 5
2. Formally enroll into the course
3. Send an E-Mail to NL@ise.tu-berlin.de with your top 3 topic wishes **until 15.5.2023 23:59**. Please send only one mail per group, but include your group members in CC.

We announce project matches on the **16.5.2023**

There are no guarantees to get a topic of your top 3, as each topic will only be assigned to three groups at most.

Cost and Performance Management

How can you strike a cost and performance balance in your application?

Task:

Work out different likely workload scenarios for your application and evaluate at what cost the as-is deployment can respond.

1. What are the costliest workloads and why?
2. Refactor the application to reduce cost for all scenarios while maintain performance.
3. Create means to observe costs in your application to stay ahead of workload changes that tip the cost/performance balance you found.

Architecture:



Online Boutique

<https://github.com/GoogleCloudPlatform/microservices-demo>



GameOn!

<https://github.com/gameontext>



Serverless Espresso [AWS]

<https://github.com/aws-samples/serverless-coffee>

Reading Material:



https://link.springer.com/chapter/10.1007/978-3-031-07481-3_5



https://link.springer.com/chapter/10.1007/978-3-319-69035-3_48

Energy Efficiency and Sustainability

Can you assess the energy efficiency and sustainability of your cloud application, if so, how?

Task:

Work out different likely workload scenarios for your application and evaluate the current CO₂ equivalent emissions.

1. What are the major factors that drive the applications' CO₂ footprint?
2. Refactor the application to adjust CO₂ emissions to each scenario.
3. Showcase how engineers can keep track of CO₂ emissions for all future development.

Architecture:



Tea Store

<https://github.com/DescartesResearch/TeaStore>



Auction Website

<https://github.com/JarroldMalkovic/auction-website>



Liberty Bikes

<https://github.com/OpenLiberty/liberty-bikes>

Reading Material:



<https://doi.org/10.48550/arXiv.2111.00364>



<https://doi.org/10.48550/arXiv.2201.11631>

Resilience / Fault Tolerance

How can you improve the resilience of your system?

Task:

Determine failures and fault scenarios that are likely in the as-is system.

1. What types of faults and failure scenarios does the architecture already address?
2. Establish and implement an approach to address missing fault or failure scenarios.
3. Implement means to showcase the new resilience fault tolerance of your system.

Architecture:



SockShop

<https://github.com/microservices-demo/microservices-demo>



GameOn!

<https://github.com/gameontext>



Serverless Espresso

<https://github.com/aws-samples/serverless-coffee>

Reading Material:



<https://doi.org/10.1109/IC2E52221.2021.00022>



<https://ieeexplore.ieee.org/document/8804433>

Performance and Elastic Scalability

How can you address both performance and scalability according to the actual workload?

Task:

Work out different workload scenarios for your application and evaluate the performance as-is.

1. What are the performance bottlenecks and why?
2. Refactor the application to improve performance and allow elastic scalability.
3. Create means to observe performance in your application and dynamically react on changing workloads.

Architecture:



Robot Shop

<https://github.com/instana/robot-shop>



SockShop

<https://github.com/microservices-demo/microservices-demo>



GameOn!

<https://github.com/gameontext>

Reading Material:



<https://doi.org/10.5220/0009792702040215>



<https://doi.org/10.1007/s10723-021-09597-5>

Privacy and Transparency

How can you enhance the transparency of personal data being processed in your system?

Task:

Find out (or define) which processing activities (compute, storage, and/or network) contain personal data.

1. Which processing activities can be captured? How (e.g., through observability)?
2. Refactor the application to improve the level of transparency with a reasonable report, inventory, dashboard or alike.
3. Create means to determine the computational and/or organizational overhead being introduced by your solution.

Architecture:



Online Boutique

<https://github.com/GoogleCloudPlatform/microservices-demo>



AstronomyShop

<https://github.com/open-telemetry/opentelemetry-demo>



RobotShop

<https://github.com/instana/robot-shop>

(you can also re-brand the shops to a more realistic or interesting scenario from a privacy perspective)

Reading Material:



https://link.springer.com/chapter/10.1007/978-3-030-99100-5_10 [DevPrivOps]



<https://ieeexplore.ieee.org/abstract/document/9284240> [Kunz et al.]



<https://dl.acm.org/doi/pdf/10.1145/3442188.3445925> [TILT]

Example: Reliability

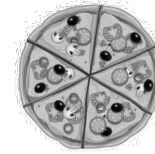
How can you improve the reliability in a pizza system?

Task:

Determine the processes that likely results in fault in the as-is system.

1. Establish and implement an approach to address failure scenarios.
2. Propose validation strategies to show the effectiveness of your approach.

Architecture:



Luigi's Pizzeria

Reading Material:



<https://doi.org/10.1016/j.jcss.2013.02.006>



<https://ieeexplore.ieee.org/document/8804433>

Luigi's Pizzeria



Problem Identification

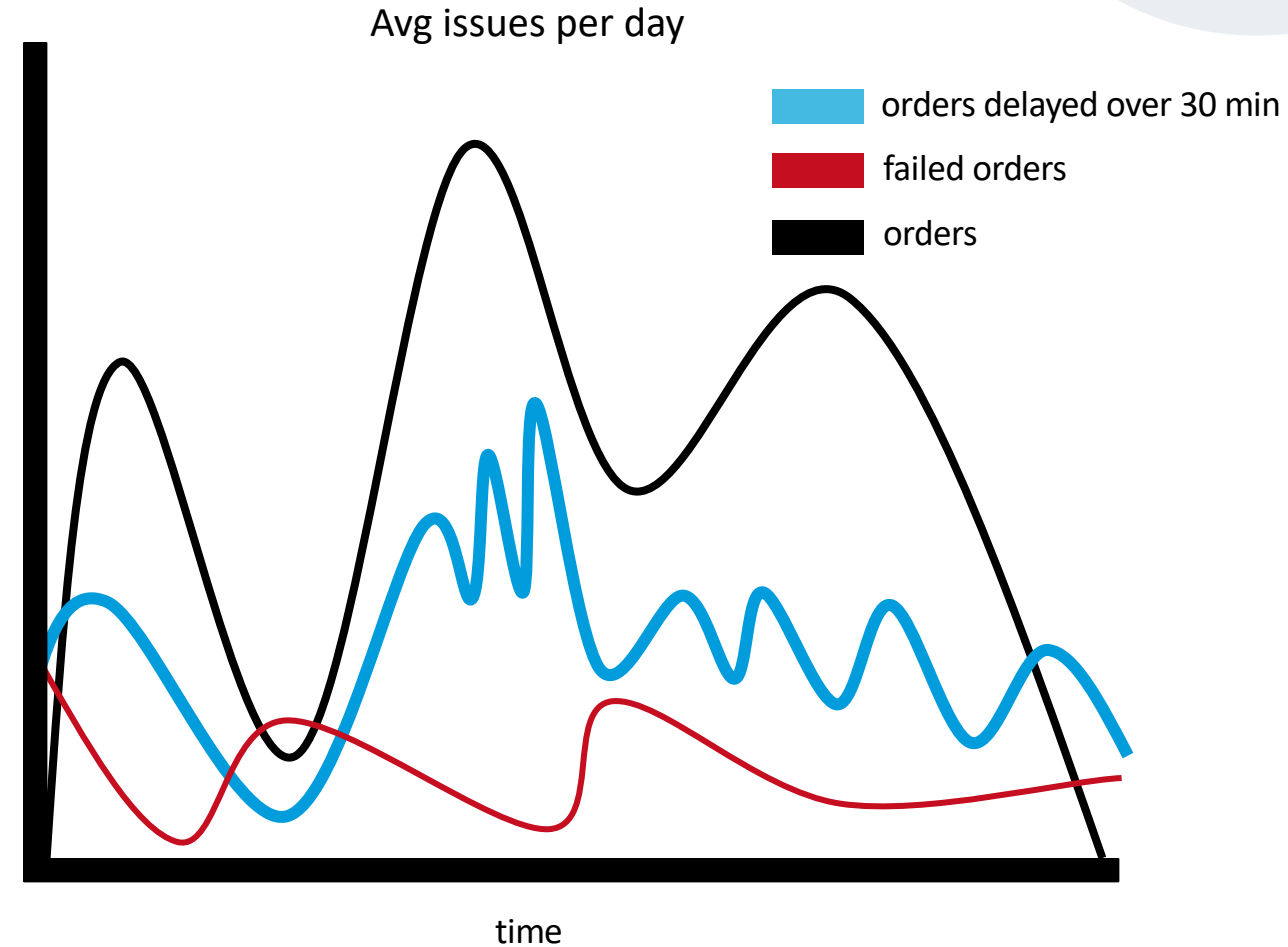
In 12% of deliveries, customers complain that their pizzas are late, and in 6% of deliveries, the pizza never arrives.

The issues are distributed across all times of the day.



Observation

- delays related to order volume
- delays fluctuate during peak deliveries
- failures happen during order downtime



Related Work



A survey on reliability in distributed systems

Waseem Ahmed*, Yong Wei Wu

Department of Computer Science and Technology, Tsinghua University, Beijing, China

Ahmed and Wu (2013)

- Review of reliability methodologies
- Need for observation and prediction

Root Cause Analysis of Noisy Neighbors in a Virtualized Infrastructure

Hedi Bouattour	Yosra Ben Slimen	Marouane Mechtri	Hanane Biallach
<i>Orange Labs</i>	<i>Orange Labs</i>	<i>Orange Labs</i>	<i>Orange Labs</i>
Chatillon, France	Chatillon, France	Chatillon, France	Chatillon, France
hedibouattour2010@gmail.com	yosra1.benslimen@orange.com	mechtri.marwen@gmail.com	hanane.biallach@orange.com

Bouattour et al. (2020)

- Elaborate phenomena of noisy neighbors
- Employ supervised learning for anomaly detection

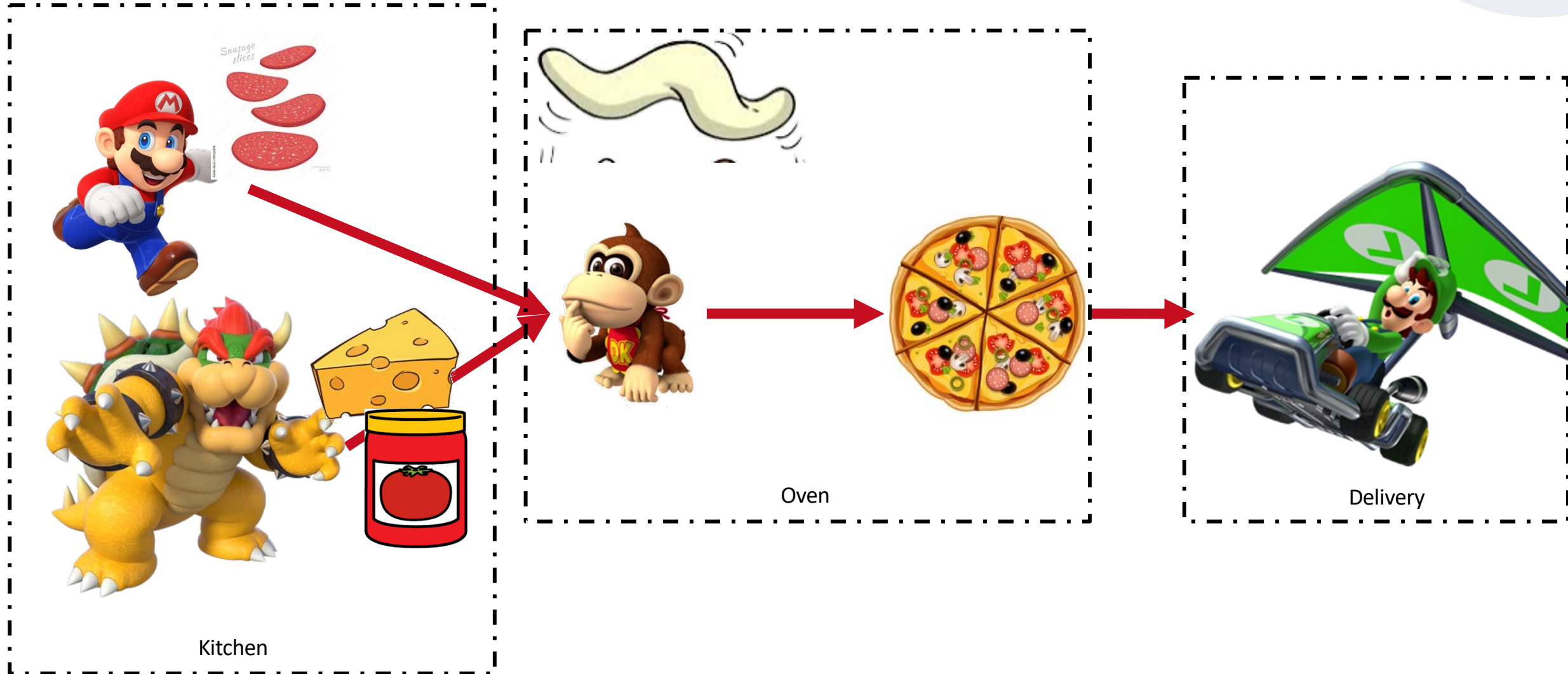
Horizontal and Vertical Scaling of Container-based Applications using Reinforcement Learning

Fabiana Rossi, Matteo Nardelli, Valeria Cardellini
Dept. of Computer Science and Civil Engineering, University of Rome Tor Vergata, Italy
{f.rossi,nardelli,cardellini}@ing.uniroma2.it

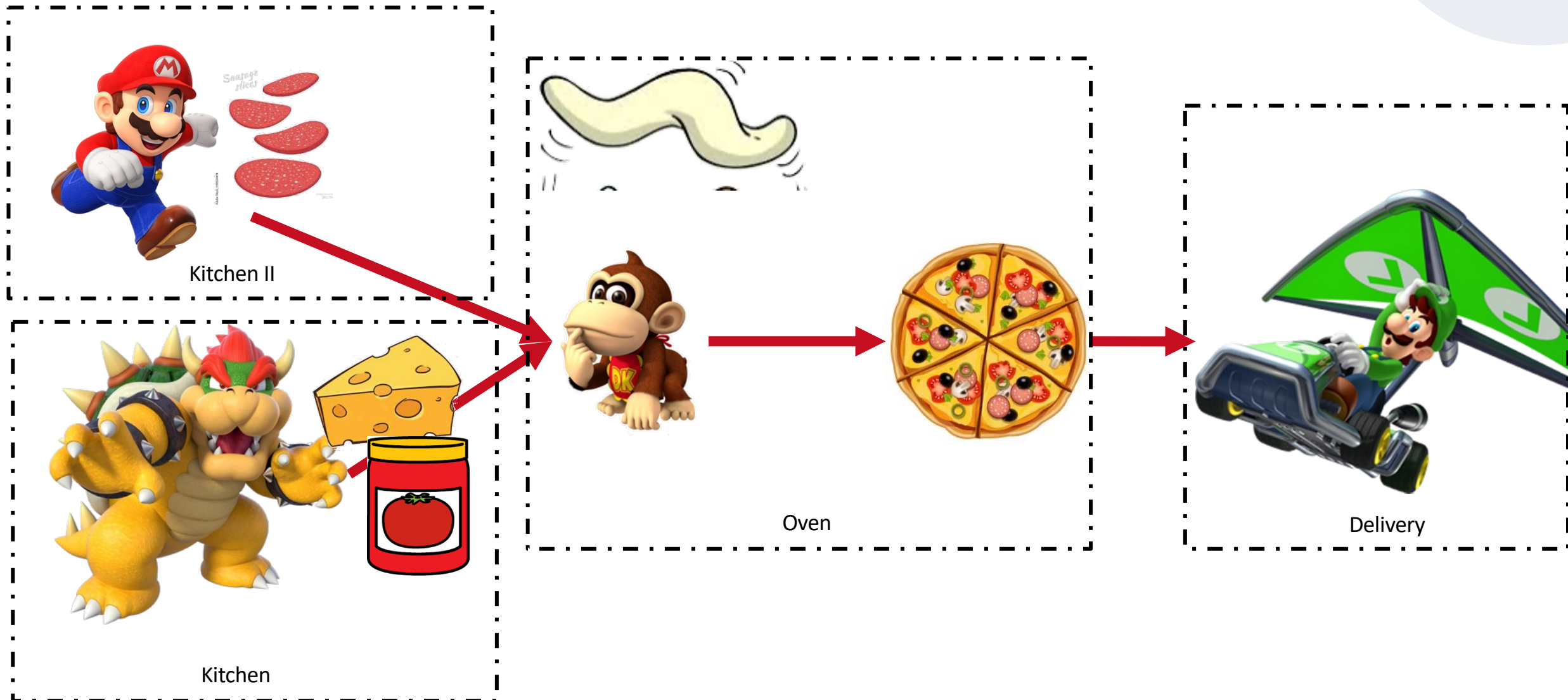
Rossi and Nardelli (2019)

- Address scalability
- Learn elasticity policies

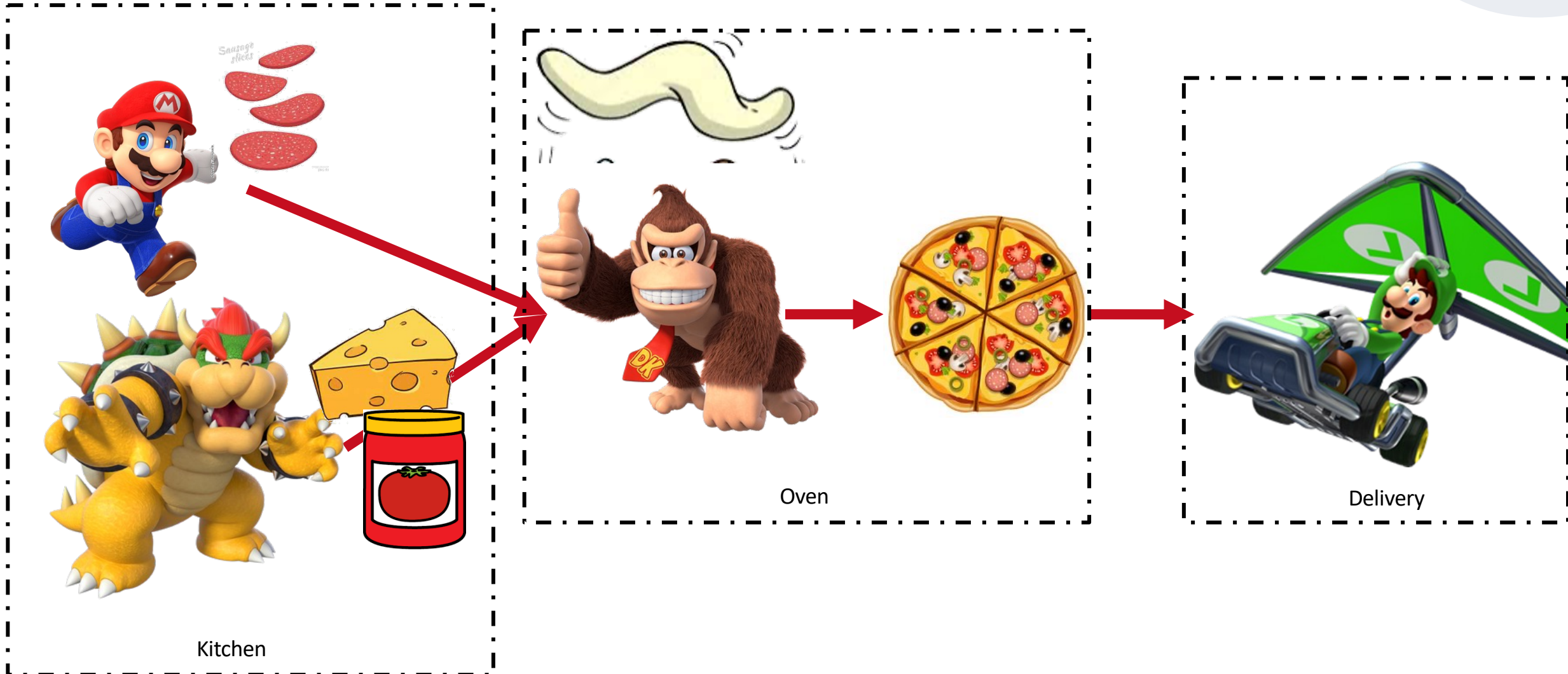
As-Is System



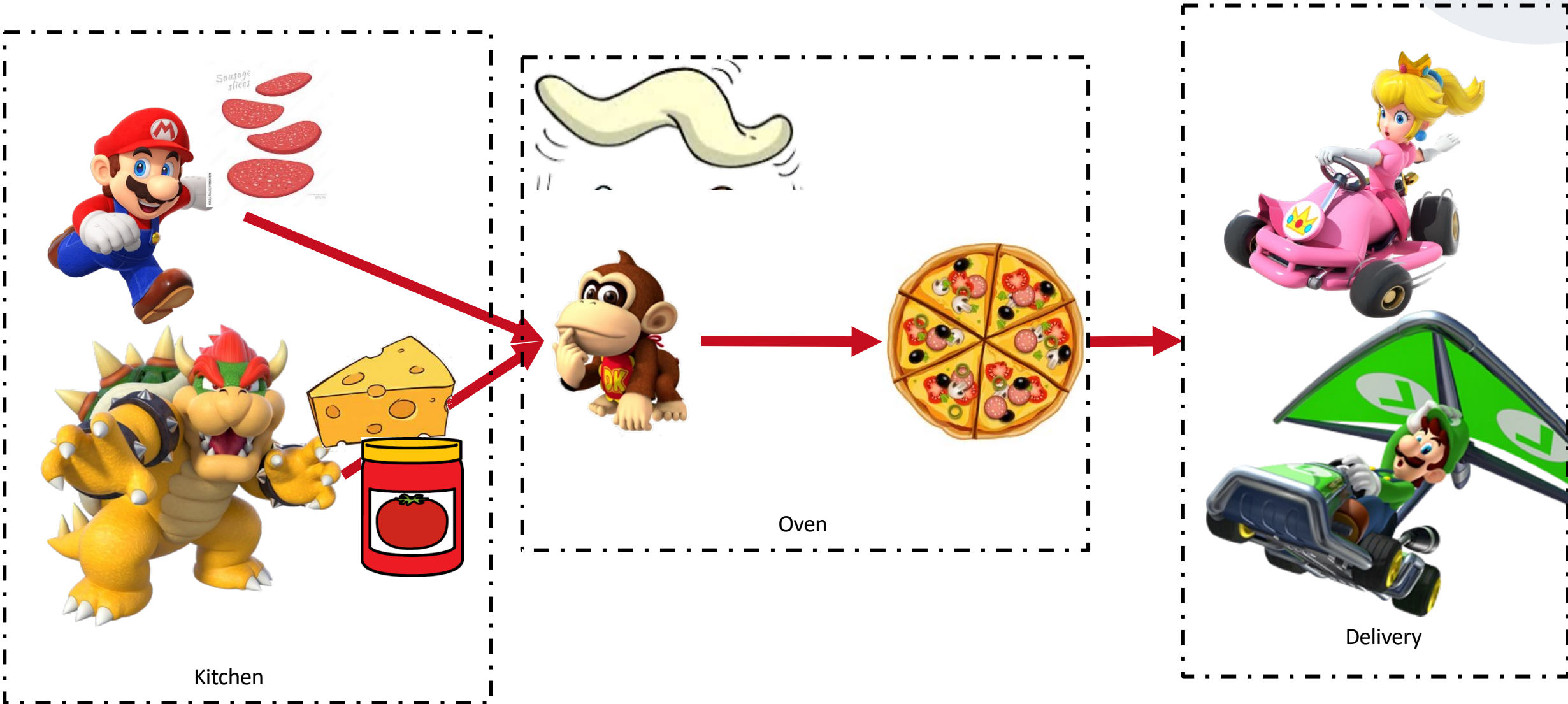
Approach (1/4) – Second Kitchen



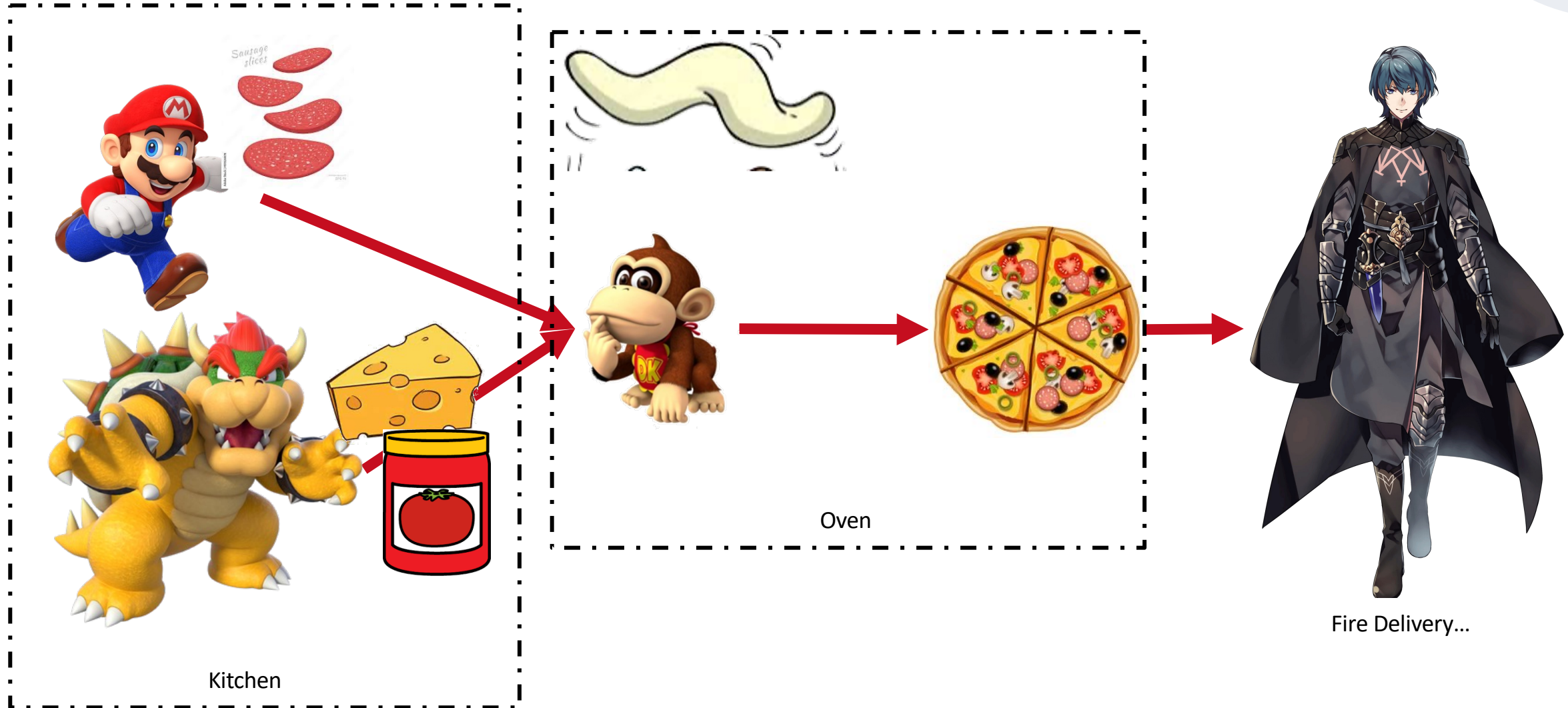
Approach (2/4) – Resizes Donkey Kong



Approach (3/4) – Scale-Up Delivery



Approach (4/4) - Outsource Delivery



Validation Approach

Observe Bowser and Mario's use of the kitchen
(are they fighting for space?)

See how Baby-Donkey Kong manages to keep up
during peak times

Observe the time of production to the time of
delivery for during peak times

Important Dates

Deadlines:

- Registration: **15/05/2023, 23:59 on Moses (MTS)**

Portfolio examinations will take place on:

- **15/05/2023:** formally enroll and send project preferences
- **12/06/2023:** upload Design Document + Slides
- **13 + 20 + 23/06/2023:** Presentations
- **03/07-14/07:** Oral consultations
- **25/08/2023:** Hand in reports

Examination in
presence.
Physical presence is
required!



Action Items:

- **Registration:** 15/05/2023, 23:59 on Moses (MTS)
- **Group Selection/Project Selection:** 12/05/2023 23:59 (ISIS + E-Mail)
- **First Feedback:** 02/06/2023 will be a group-individual consultation

We recommend you use the remaining time to form groups.
To help you find colleagues with similar interests,
we put signs with the topics on the walls.

- 1: Cost and Performance Management
- 2: Energy Efficiency and Sustainability
- 3: Resilience / Fault Tolerance
- 4: Performance and Elastic Scalability
- 5: Privacy and Transparency

CNAE Schedule (still preliminary)

